

# Robust Method of Detecting Moving Objects in Videos Evolved by Genetic Programming

Andy Song  
RMIT University  
Victoria, Australia  
andy.song@rmit.edu.au

Danny Fang  
RMIT University  
Victoria, Australia  
dfang@cs.rmit.edu.au

## ABSTRACT

In this paper we investigated the use of Genetic Programming (GP) to evolve programs which could detect moving objects in videos. Two main approaches under the paradigm were proposed and investigated, single-frame approach and multi-frame approach. The former is based on analyzing individual video frames and treat them independently while the latter approach consider a sequence of frames. In the single-frame approach, three methods are investigated including using pixel intensity, pixel hue value and feature values. The experiments on Robosoccer field show that GP could detect the target under different lighting conditions and could even handle arbitrary camera positions. Although there was no domain knowledge had been provided during evolution, GP was able to produce moving object detectors that were robust and fast.

## Categories and Subject Descriptors

I.5.4 [Pattern Recognition]: Applications; I.2.10 [Vision and Scene Understanding]: Video analysis

## General Terms

Algorithms, Design, Experimentation

## Keywords

Genetic Programming, Video Analysis, Object Detection, Tracking, Motion Detection, Real Time, Robosoccer

## 1. INTRODUCTION

Automatic detection of moving object in video streams has great importance for many vision applications such as surveillance systems, robot vision systems, traffic control systems and auto-pilot/driving systems. The need is growing fast with the increased affordability and popularity of video capture devices. The goal of these vision systems are usually recognizing objects on the scenes and following the

moving objects. The core part of these systems is a flexible, reliable and fast video image analysis methodology which can process video signal in real-time.

The goal of our study is to utilise Genetic Programming (GP) to address these tasks. We use the robot soccer environment as a testbed. The conventional robosoccer systems are sensitive to environment changes and require recalibration when change occurs. Also domain knowledge such as colour of the ball, colour of the field need to be provided to the system. In contrast we are aiming to use GP to create moving object detectors which require less specific domain knowledge and are able to handle variations. Appropriate methodology of evolving such detectors is investigated, especially the suitable representations. The generated detector were tested on real-time video inputs.

## 2. BACKGROUND

Genetic Programming has been applied to still object detection soon after Koza introduced it as a powerful problem solving method [11, 13]. It has been used in detecting various objects such as tanks, coins and faces. GP has also been used for multiple object detection [14]. These works showed that GP could be used as a tool for developing domain-independent learning/adaptive approaches for detecting small objects of multiple classes in static images. [4] used GP based object detection for mining vehicles based on extracted features. The GP object classifiers in this study was able to provide a certain degree of robustness. The evolved object detectors that could handle different lighting conditions. GP has also been extended to tracking faces in videos based on image features [10].

### 2.1 Object Detection in Robot Soccer

RoboCup soccer is an active competition among enthusiastic AI researchers [8]. Players need to be able to locate the ball in real-time. Thresholding for segmentation is the most popular object detection technique for object detection and localisation in robot soccer. It is largely colour based. Thresholds in hue histograms performs well on noise and illumination variations [12]. However they are fragile and need manual adjustment to suit its operational environment. [2] created a system that segments objects based on colour. It also required manual adjustment when conditions change. To create more robust thresholding systems, learning methods have been used provide solutions that can handle changes in illumination and noise. [5] used a simple EA to evolve a robust colour classifier for robot soccer by chrominance space transformation. [1] used a neural net-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.  
Copyright 2008 ACM 978-1-60558-130-9/08/07...\$5.00.

work to determine the thresholds for object classes in a hue histogram. [3] utilized decision trees to perform the same task of thresholding for classification. The above methods incorporated pre-processing of frames to filter out noise, illumination and colour variations. Other techniques used include active contours [7] and the use of laser range finders for scan line matching [6]. These existing techniques almost always rely heavily on specific domain knowledge.

### 3. METHODOLOGY

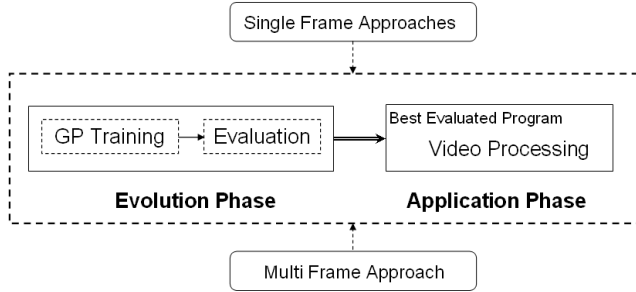


Figure 1: Overview of the Investigation

The overview of our investigation is shown in Figure 1. It can be split into two halves known as the *evolution phase* and the *application phase*. The evolution phase is the process to evolve and evaluate GP programs. The application phase is using the evolved classifier to process unseen videos for detection of moving objects. Based on the same framework for evolution and application, we propose and investigate two approaches: single-frame approach and multi-frame approach. Other than these two kinds of approaches, the experiment procedure, the GP settings and the method of detecting objects based on evolved programs are consistent cross our investigation.

#### 3.1 Data Generation

To train GP programs as an object detector, a set of examples with correct class labels should be provided. However a single-frame from video is usually 384 x 288 pixels which is too big to be used as inputs of GP programs. So small cut-outs are sampled from frame images by a sweeping window. The window size is 32 by 32 pixels. Cut-outs containing target object, soccer ball in our case, and cut-outs containing only the background are marked as different classes. This set of cutouts is split into two parts to be used for training and evaluation respectively.

#### 3.2 Image Representations

Four representations were investigated in this study: pixel intensity value, pixel hue value, feature values and motion plane. The first three (intensity, hue and feature) are under the single-frame approach while the fourth is in the multi-frame approach.

##### 3.2.1 Pixel Intensity Value

This representation directly uses the intensity values of each pixel in a cutout image, normalized to range of 0 to 100. Intensity was chosen because it is the most basic representation and is readily available. The color information

is discharged so this representations would be insensitive to colour changes. For one cutout image, the number of pixels on that images is number of data points in this representation.

##### 3.2.2 Pixel Hue Value

This representation records the hue value of each pixel. The value is calculated by converting RGB values of the pixel and then applying a sine function. So the values should be between -1 and 1. The values of  $Hue = 0$  and  $Hue = 359$  would not far away since visually they are in similar color. The choice of hue is due to its popular use in robosoccer vision systems. In this representation the number of data points for an image is also equal to the number of pixels on that image.

##### 3.2.3 Feature Based Representation

Unlike the above two pixel based representations, feature representation is region based. An image is divided into five regions as shown in Figure 2: the left-top quarter, the left-bottom quarter, the right-top quarter, the right-bottom quarter and the middle quarter. A set of twelve features can be calculated from there five regions:

- Features 1-5: the average intensity values of the pixels in each of the corresponding regions.
- Feature 6: the magnitude of the first four features minus four times the fifth feature.
- Features 7-12: are averages of different pairings of the first four features.

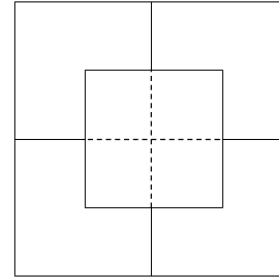


Figure 2: Five Regions for Generating Features

This representation is similar to that in other GP-based objection detection studies such as [14]. Image features could more likely capture characteristics of an image. The feature of an image containing a ball in the center might be significantly different to that of not-ball images. It would dramatically reduce the search space of GP. Furthermore the data points are much less in this representation. In our case, images always have 12 feature values despite their sizes.

##### 3.2.4 Motion Plane

The above three representations are for single frame images. GP programs using these representation treat video frames independently. To preserve the coherence of consecutive frames, we used a multiple frame approach by which intensity values were combined from several frames into one frame, called *motion plane*. The formula for calculating a

motion plane is shown below:

$$MP(x) = \frac{\sum_{i=1}^{n-1} ((x_i - x_{i-1}) \times (n - i))}{n}$$

Where  $x$  is one position in the video. There are  $n$  consecutive frames, frame 0 is the most current one and frame  $n - 1$  is the oldest in the queue.  $n - i$  acts as a weight so that changes in more recent pairs of frame will have more influence on the value of the motion plane. The sum of weighted differences is divided by  $n$  to produce the average change value for one pixel in the motion plane. By this representation the change over a sequence of frames, or motion information, is stored. This representation is suitable for catching moving objects on a stationary background, not on moving background.

### 3.3 Evolution Phase

This part describes the details of evolution phase including GP settings, runtime parameters, data generation, as well as the process of selecting programs for the application phase.

#### 3.3.1 Function Set

Function	Return Type	Arguments
+	Double	Double, Double
-	Double	Double, Double
×	Double	Double, Double
/	Double	Double, Double
=	Boolean	Double, Double
<	Boolean	Double, Double
>	Boolean	Double, Double
if	Double	Boolean, Double, Double
between	Boolean	Double, Double, Double

Table 1: Function Set

Functions are the internal nodes of a GP tree. The function set used (Table 1) includes four basic arithmetic operators **add**, **subtract**, **multiply** and **divide** that return doubles and three comparative operators **equals**, **less than** and **greater than** which return booleans. Both of the above functions take in two doubles as inputs. There are two additional functions that takes three inputs **if** and **between**. The **if** function takes in a boolean and two doubles, if the boolean is true then the first double is returned else the second double is returned. The **between** function takes in three doubles as input, if the second double is between the first and the third then it will return true else it will return false.

#### 3.3.2 Terminal Set

Terminals are the leaf nodes of a GP tree. The terminal set (Table 2) contains two different types of terminals, the first(**drand**) being a random double that is generated by the GP and the other is attributes. **Attribute(Att)** is the input of a GP programs. It could be normalized intensity value, sine of hue, some feature values or one point from the motion plane, depends on which approach is being used.

Terminal	Return Type
drand	Double
Att[x]	Double

Table 2: Terminal Set

Population Size	200
Maximum Depth	12
Minimum Depth	2
Generations	150
Mutation Rate	0%
Crossover Rate	90%
Elitism Rate	10%

Table 3: GP Runtime Parameters

#### 3.3.3 Fitness Function, Termination Condition and Run Time Parameters

We used the accuracy of a GP program as fitness measure which is the percentage of small cutouts in training being recognized correctly. There are two termination conditions. The first one is when the training accuracy reaches 100% while the second is when the number of specified generations has been reached.

The GP runtime parameters are listed in Table 3. To generate the initial population the ramped half-and-half method was chosen as it could provide a wide variety of initial program trees [9]. The mutation rate was set to 0% as works done on similar problems utilised mutation rates that were either very low or zero. Such a parameters choice is based on the parameter choices of our other related works such as recognizing texture pictures, analyzing X-ray images. This parameter choice is also consistent with that in image related GP works reported by other researchers.

#### 3.3.4 Training/Evaluation

The cutouts generated were split into two folds for training and evaluation. There were always an equivalent number of positive and negative cases in each fold. Positive cases are all relatively similar: the ball located around the center of the cutout with some variations of background. Negative cases contain a large number of varieties such as the green field, part of the goal, part of the boundary lines. The data fold for training has two thirds of the cutouts while the fold for evaluation has the rest. For each generation of the training, the best GP programs are passed to evaluation. When the evolution process finishes, the best performing programs during the evaluation are then selected for the next phase.

### 3.4 Application Phase

In this phase an evolved program is deployed to process videos. The basic procedure is illustrated in Figure 3. The main steps are: retrieving the current frame from video input (Step 1), using a sweeping window to sample small cutouts of which the size is consistent with that in evolution phase (Step 2), classifying these cutouts by evolved GP program (Step 3), assembling the labeled cutouts back to a frame (Step 4) and pushing the processed frames continuously as video output (Step 5).

In Step 2 every pixel is sampled and most of the pixels are sampled several times within windows at different locations because there are overlaps between adjacent window positions. Similarly one pixel might be classified multiple times within different cutouts. A voting strategy is used in step 4 to deal with such situation. For example if a pixel has been classified as “ball” more times than being classified as “not-ball”, then this pixel is considered a not-ball pixel in the final output.

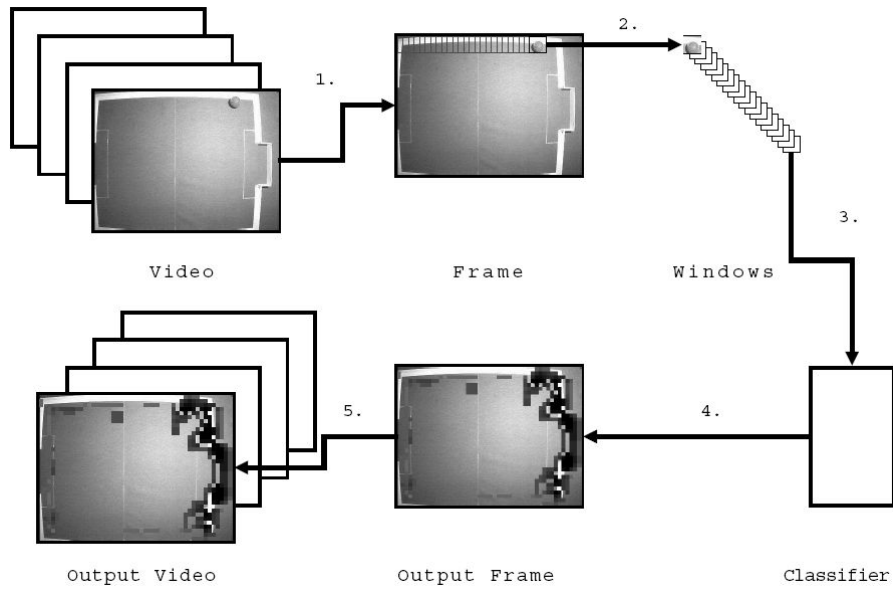


Figure 3: Object Detection Based on Evolved GP Program

The degree of overlap in Step 2 is adjustable. Larger overlap will generate smoother output, but result larger amount of cutouts, hence requires more processing time. One can use this parameter to balance the computation cost and output quality. The overlap used in these experiments is always 4 pixel.

In the following experiments, the video input for test was streamed at 25 frames per second. The frame size does not have to be the same in the evolution phase. Two types of tests were conducted, the first being variations in illumination while the second being variations in ball distance and camera positions. The resulting accuracies and false positive rates were recorded from the video output.

## 4. EXPERIMENTS

This section presents the results of evolving based on the above GP settings according to four representations. The three representations in single-frame approach were provided with a total of 3106 cut-outs generated from 360 frames, 2070 used in training and 1036 used in evaluation. In multi-frame approach 120 groups of frame sequence were collected and produced 2652 cutouts, 1768 for training and the rest 884 for evaluation. All these frames were taken under the same condition. The lighting conditions and the camera positions were consistent. There were three Halogen lights over the robosoccer field.

### 4.1 Pixel Intensity Values

Figure 4 illustrates the evolutionary process based on intensity representation. The training and evaluation accuracies in the graph are the average over five runs. It can be seen that the evaluation accuracy curve is always close but below the curve for training accuracy. That indicates that GP was able to improve the accuracy from 70% to above 90%, and there was no significant overtraining observed.

Figure 5 shows the test results generating by the evolved program under different lighting conditions. Some sample

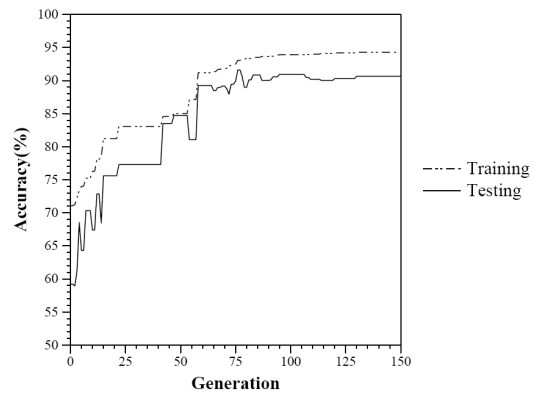


Figure 4: Evolution Process (Pixel Intensity)

output frames are listed. The “Accuracy” column lists the percentage of the pixels being correctly marked during the test. The “FP Rate” is the false positive rate which indicates the percentage of not-ball pixels being marked as ball. Accuracies and FP rates in the figure are average values under each lighting conditions.

It can be seen that the ball could always be found in the videos despite the lighting changes. The accuracy of the GP program was able to maintain around 95%. Light changes only marginally affected the GP accuracy. There were some false positives, especially along the edge. That is possibly due to the lack of examples. There are much more cutouts of fields, balls than the cutouts of edges. The drop of accuracy caused by lighting changes might due to the increasing false positives.

The false positives could be reduced by post-processing such as analyzing the shape or size of the marked areas. Post-processing was not introduced here because the aim of this study is the GP methodology rather than improvement.

Output Frame	Accuracy	FP Rate
		
Three Halogen Lights	96.15%	2.59%
		
Two Halogen Lights	95.33%	2.66%
		
One Halogen Light	94.92%	5.08%

Figure 5: Performance on Video (Pixel Intensity)

## 4.2 Pixel Hue Values

Similar to Figure 5, Figure 6 presents the performance of using hue representation. The training accuracy was slightly higher than that of intensity approach. However its performance on video input was poor. Although the ball could be marked under all runs, there were significant amount of false positives. Actually they were the reason for majority of the errors. According to Figure 5, the error rate under three lights was 10.71% but the FP rate is 9.53%, count for 89% of errors.

Despite the poor performance, the GP program showed its capability of handle variations. Its performance under three lighting conditions were rather similar.

## 4.3 Feature Values

The performance of feature based approach is even worse. The evaluation accuracy was just around 80%. During the test on videos the program resulted even more false positives although the ball could always ben found. The poor performance was not affected much by lighting changes either. The accuracies under three light conditions were all around 85% while the false positive rates were all around

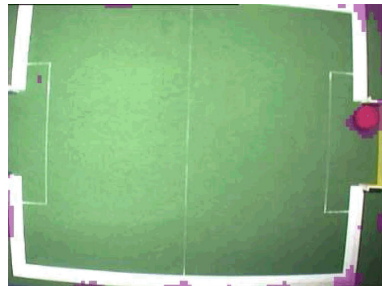
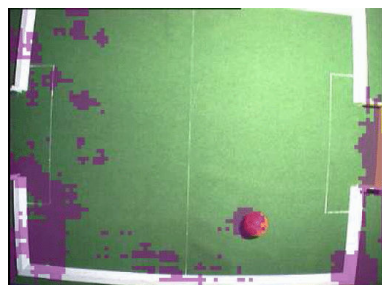
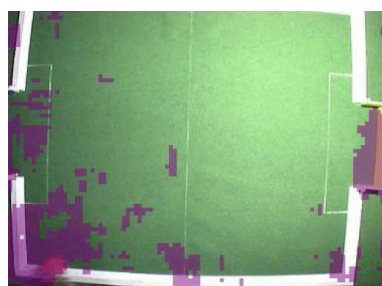
Output Frame	Accuracy	FP Rate
		
Three Halogen Lights	89.29%	9.53%
		
Two Halogen Lights	91.31%	8.04%
		
One Halogen Light	88.48%	10.48%

Figure 6: Performance on Video (Pixel Hue)

14%. The output video frames are not presented since they add little value.

## 4.4 Motion Plane

Output Frame	Accuracy	FP Rate
Three Halogen Lights	98.18%	0.28%
Two Halogen Lights	98.59%	0.16%
One Halogen Light	98.18%	0.83%

Figure 7: Performance on Video (Motion Plan)

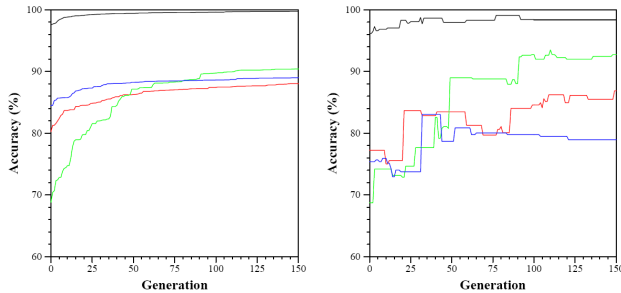
The performance of multi-frame approach is stronger than that of single-frame approaches. Its training and evaluation accuracies were above 98%. It has little false positives when applying on videos. The accuracies on video frames were also 98%, as shown in Figure 7.

## 5. DISCUSSION

The observation from the experiments and further investigation are discussed in this section.

## 5.1 Comparison

Comparing the performance of the four representations in application phase, it can be seen that the motion plane and intensity methods can achieve higher accuracy, or less false positives. Feature approach seems be the last choice among the four. This might due to the twelve features described before are not very suitable for the task. However finding more appropriate feature set itself is a search task. Other three representations are all readily available.



**Figure 8: Training and Evaluation Accuracies of Four Approaches**

The ranks are also reflected in the evolution phase. The trend of training and evaluation accuracies of the four approaches during the evolution are shown in Figure 8. The top lines are from motion plane approach and the bottom lines are from feature approach.

## 5.2 Arbitrary Camera Positions

The evolved programs generated in experiments of last section were applied on video taken from new camera positions to further test their robustness. Changes in camera positions would introduce dramatic variations including overall intensity, overall hue and background. Furthermore this might bring in unknown scenes or new objects. Additionally it causes size variation. In the above experiments, the camera was always steadily hanging over the soccer field. The size of the ball appeared roughly the same with this camera position. However if the angle changes, then ball size would be affected by its distance to the camera.

Figure 9 shows the test on the same program used in Figure 5. The ball can be found although the sizes are different from these three frames. Note this GP program had never seen the ball at a different distance. It still can pick up the ball without prior knowledge. Never before seen objects including the power cable and the chair were rightfully ignored by the classifier as well as the new backgrounds of carpet and wall. Surprisingly the number of false positives that were detected were significantly lower than that of the original camera position.

Figure 10 depicts the results of the hue based classifier. Once again the classifier was able to find the ball independent of camera angle or ball distance. Despite it having a greater of false positives, the classifier was able to have better coverage over the ball on the three new positions. It was less successful in ignoring new backgrounds and objects especially on the boundary between the carpet and the wall.

Figure 11 is the results of the classifier using motion plane representation. It performed very well in alternate conditions. Alternate camera angles and ball distance seemed

no impact on its performance. This is expected because it works on motion information which should not be affected by stationary objects and background.

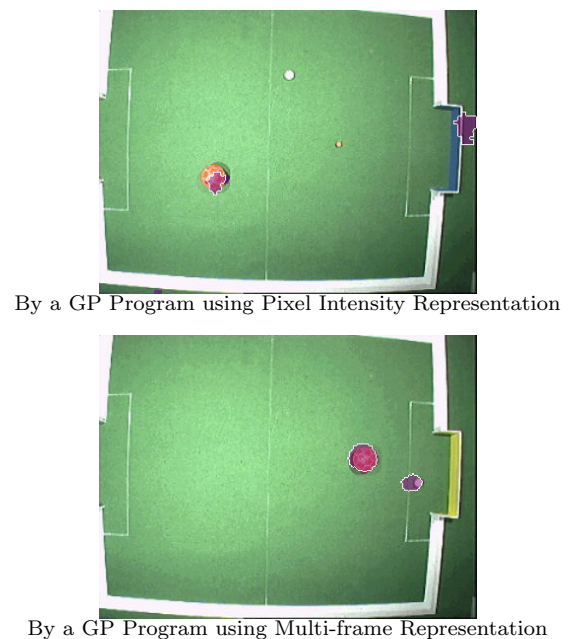
Due to its previous poor performance, classifier based on features was not studied.

## 5.3 Speed Advantages

No matter under what lighting conditions or camera positions, all the above tests on video achieved real-time performance. No delay was observed when running on an ordinary Apple MacBook Pro 2.33G with 2G RAM. This is due to the small size of evolved GP programs. In the above approaches the program varies from 58 nodes to 102 nodes. They take around 21 to 43 milliseconds to process a single frame of size  $384 \times 288$ . So handling live video at a rate of 25 frames per second is achievable by these programs. In many vision applications, the scene does not change very rapidly, so it is not necessary to process 25 frames each second. This could release more resources for tasks like pre-processing or post-processing.

## 5.4 Intensity vs. Motion Plane

The above investigation shows that intensity and multi-frame representations have good performance. Intensity approach can not only locate the moving object, but also identify it by giving its class label e.g. “ball” or “not ball”. For example, the top image of Figure 12 was generated by the same program used for Figure 5. It can only recognize soccer balls, the baseball and the golfball were not detected. In comparison the multi-frame approach can detect multiple kinds of moving objects as shown in the bottom image of Figure 12. Both moving soccerball and moving baseball were marked. However it does not know what the moving object is. Furthermore it does not recognize stationary objects. These two approaches have their own merits and drawbacks. In the further study we would combine them.



**Figure 12: Multiple Objects on Videos**

## 6. CONCLUSIONS

The overall goal of this work is to investigate how GP can be used to produce robust moving object detectors in videos. Four different methods for adapting GP in this domain are studied. This includes two major approaches known as the single-frame approach and multi-frame approach. Under the single-frame approach three data representations were investigated. They included pixel intensity based, pixel hue based and feature based methods. One representation, the motion plane, is investigated under the multi-frame approach. All these methods have been tested on unseen new frames and they were able to find the ball from video frames.

In terms of the accuracy measurement, pixel classification accuracy was used as the indicator although object detection usually does not require such precision. Despite the strict measure of accuracy the evolved GP programs can still achieve good performance. The pixel intensity based approach is proved to be the best performer out of the single-frame approaches with an accuracy rate of above 95% on video inputs. The proposed multi-frame method performed the best out of all the different methods on all the tests with an accuracy of above 98%. Both methods were able to produce real-time moving object detection solutions that performed a rate of 25 frames per second.

The evolved GP moving object detectors were tested for illumination invariance, shift invariance and size invariance. They were only trained under one set of conditions, but were still able to locate the ball under alien conditions. All single-frame GP object detectors were able to handle changes in illumination conditions at the cost in slightly increased false positive rates. The multi-frame approach proved to be able to achieve the same high levels of performance under different lighting conditions. When the GP detectors were tested under alternate camera positions they were all still able to detect the ball despite the dramatic changes in the scene. Ball size also changed with its distance from camera under different positions. The programs were still able to locate the ball.

In conclusion GP is suitable for this complex task and can evolve moving object detectors which are robust and fast. Furthermore developing such GP detectors does not require much specific domain knowledge. The framework can be easily transferred to detecting moving objects in other scenes.

The errors in this investigation were mostly due to false positives. Post-processing algorithms could be developed to analyze output frames to eliminate false positives and to determine the exact coordinates of the target object. These will be addressed in our future work.

Moreover the methodology here is new in the field and it requires further investigation such as how to cope with moving backgrounds and how to use texture information. Also our methodology should be readily applicable for detecting multiple moving objects. That will be another interesting topic of our future study.

## 7. ACKNOWLEDGEMENTS

We would like to thank Bingxuan Xie for his assistance in coding the GP video platform and in our experiments. We would also like to thank members of the Evolutionary Computing and Machine Learning (ECML) group at RMIT for contributing their thoughts and ideas for the project.

## 8. REFERENCES

- [1] C. Amoroso, A. Chella, V. Morreale, P. and Storniolo. A Segmentation System for Soccer Robot Based on Neural Networks. *Proc. of the RoboCup-99 Workshop, Stockholm*, 1999.
- [2] T. Bandlow, M. Klupsch, R. Hanek, T. Schmitt. Fast Image Segmentation, Object Recognition and Localization in a RoboCup Scenario, *RoboCup-99: Robot Soccer World Cup III*, pages 174-185, 2000.
- [3] J. Brusey and L. Padgham. Techniques for obtaining robust, real-time, colour-based vision for robotics, *Robocup Workshop, Stockholm*, 1999.
- [4] T. Burgess and V. Ciesielski. A comparison of visual object detection methods for underground mining vehicles, *Commercial in Confidence*, 1993.
- [5] T. Dahm, S. Deutsch, M. Hebbel and Osterhues, A. Robust color classification for robot soccer. *7th International Workshop on RoboCup*, 2003.
- [6] J. Gutmann, T. Weigel, B. Nebel. Fast, accurate, and robust self-localization in polygonal environments. *Proceedings of 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999.
- [7] R. Hanek, T. Schmitt, S. Buck, and M. Beetz. Fast image-based object localization in natural scenes. *2002 IEEE/RSJ International Conference on Intelligent Robots and System*, 2002.
- [8] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa and H. Matsubara. RoboCup: A Challenge Problem for AI and Robotics, *Lecture Notes In Computer Science*, pages 1-19, 1998.
- [9] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Bradford Books, 1992.
- [10] W. Smart and M. Zhang. Tracking Object Positions in Real-time Video using Genetic Programming. *In Proceeding of Image and Vision Computing International Conference*, pages 113-118, Akaroa, New Zealand, Nov, 2004.
- [11] J. Winkeler, and B. Manjunath. Genetic programming for object detection. *Genetic Programming*, pages 330-335, 1997.
- [12] G. Wyeth and B. Brown. Robust Adaptive Vision for Robot Soccer. *Mechatronics and Machine Vision in Practice*, pages 41-48, 2000.
- [13] M. Zhang. Improving Object Detection Performance with Genetic Programming. *International Journal on Artificial Intelligence Tools*, 16(5):849-873, 2007.
- [14] M. Zhang, V. Ciesielski, and P. Andreae. A Domain-Independent Window Approach to Multiclass Object Detection Using Genetic Programming. *EURASIP Journal on Applied Signal Processing*, 2003(8):841-859, 2003.



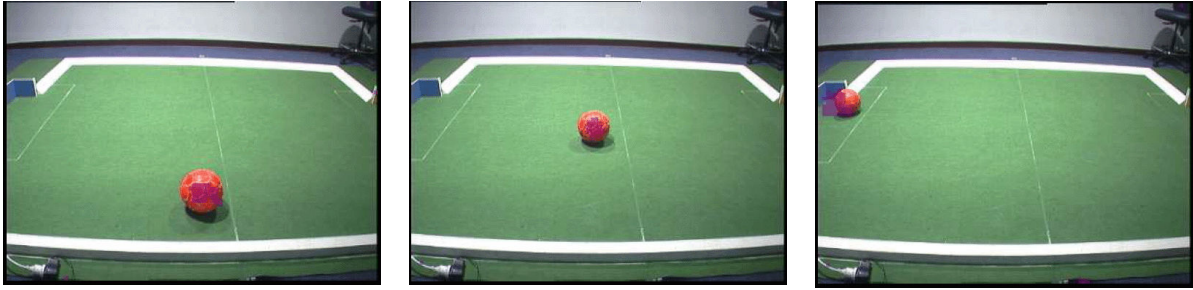


Figure 9: Test with New Camera Positions (Pixel Intensity)

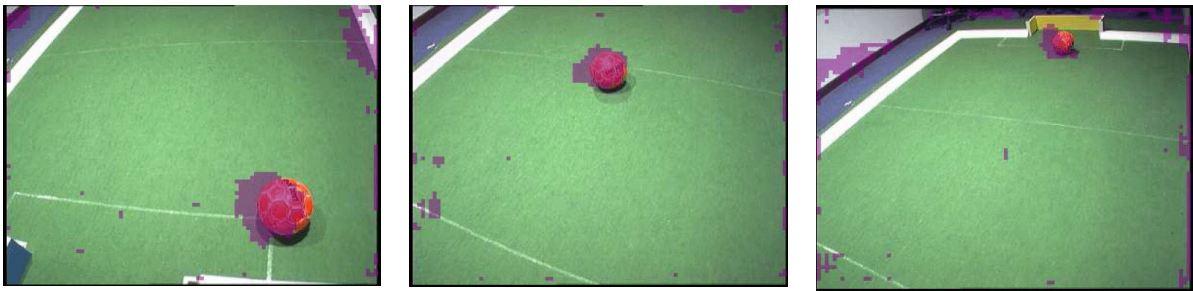


Figure 10: Test with New Camera Positions (Pixel Hue)

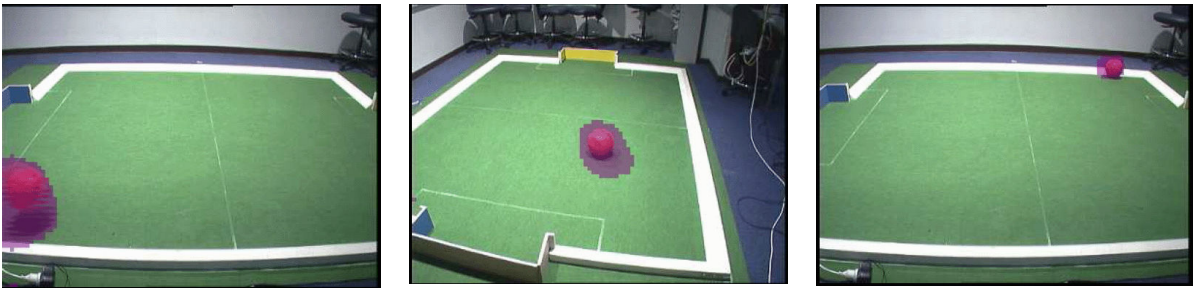


Figure 11: Test with New Camera Positions (Motion Plane)